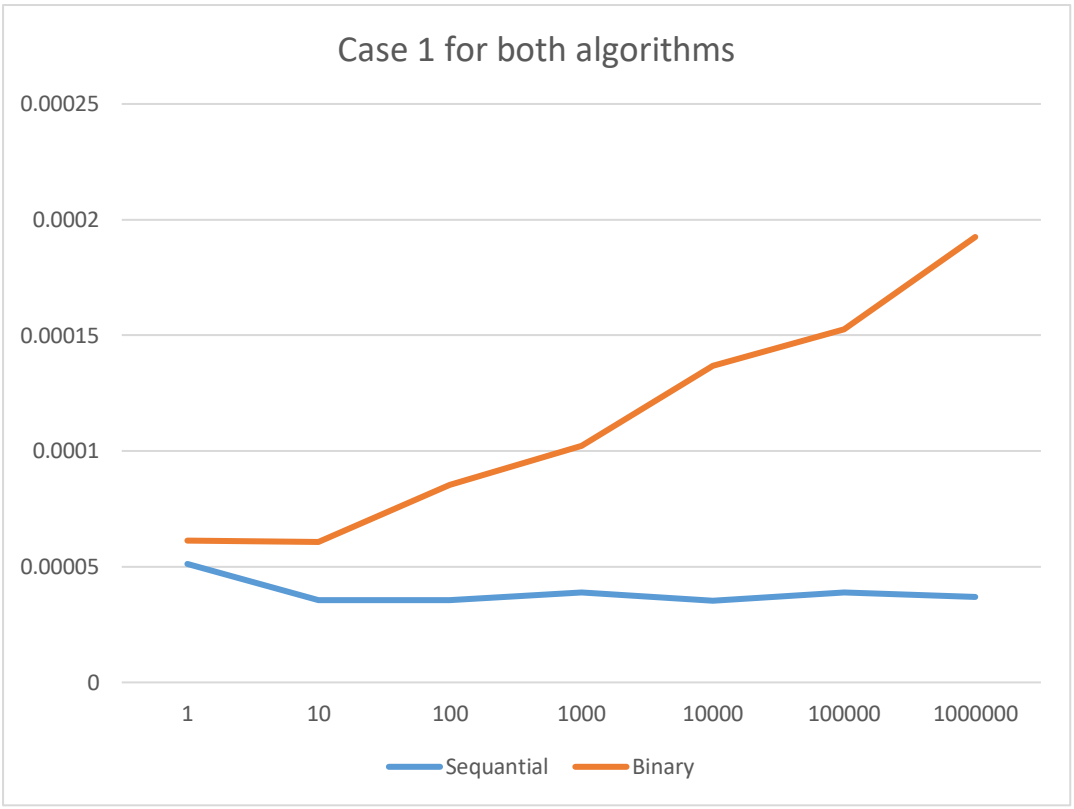
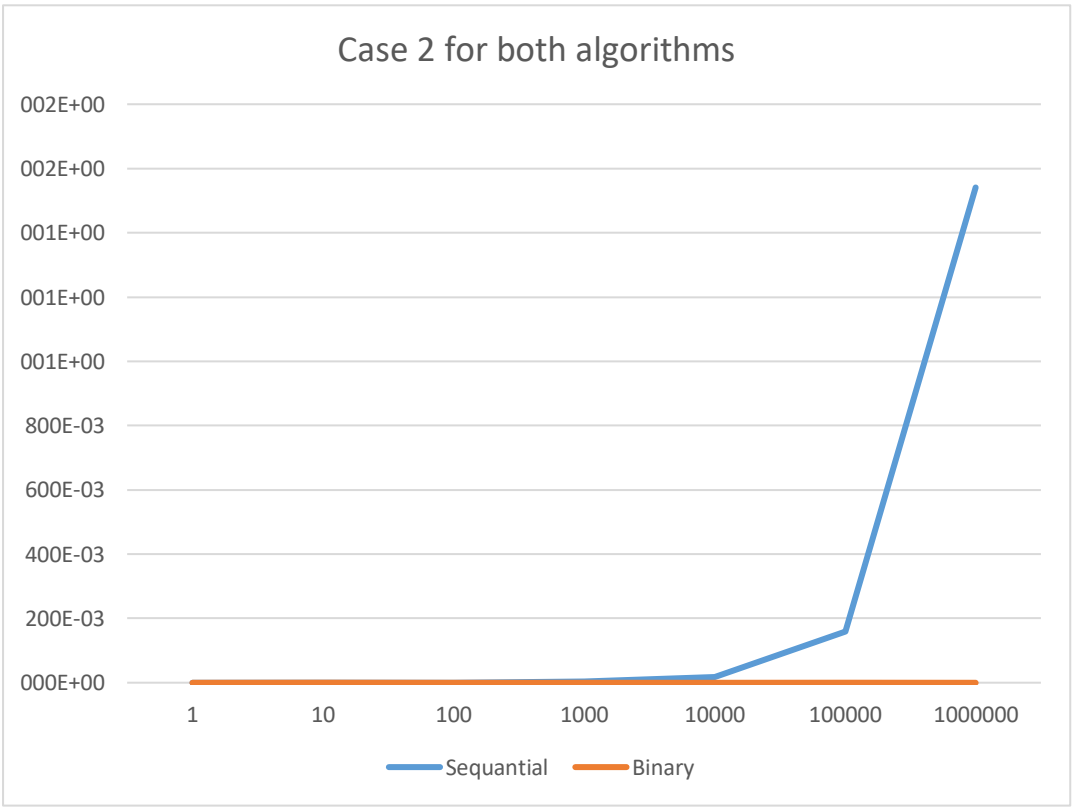
Time(ms)	Case 1	Case 2	Case 3	Case 4
N = 1	5,12e-05	5,51e-05	4,31e-05	3,75e-05
N = 10	3,56e-05	4,44e-05	5,93e-05	5,94e-05
N = 10^2	3,55e-05	2,015e-04	0,0003589	0,0003611
N = 10^3	3,9e-05	3e-03	0,005	0,006
N = 10^4	3,53e-05	1,7e-02	0,036	0,037
N = 10^5	3,9e-05	1,6e-01	0,304	0,31
N = 10^6	3,69e-05	1,541	3,106	3,094

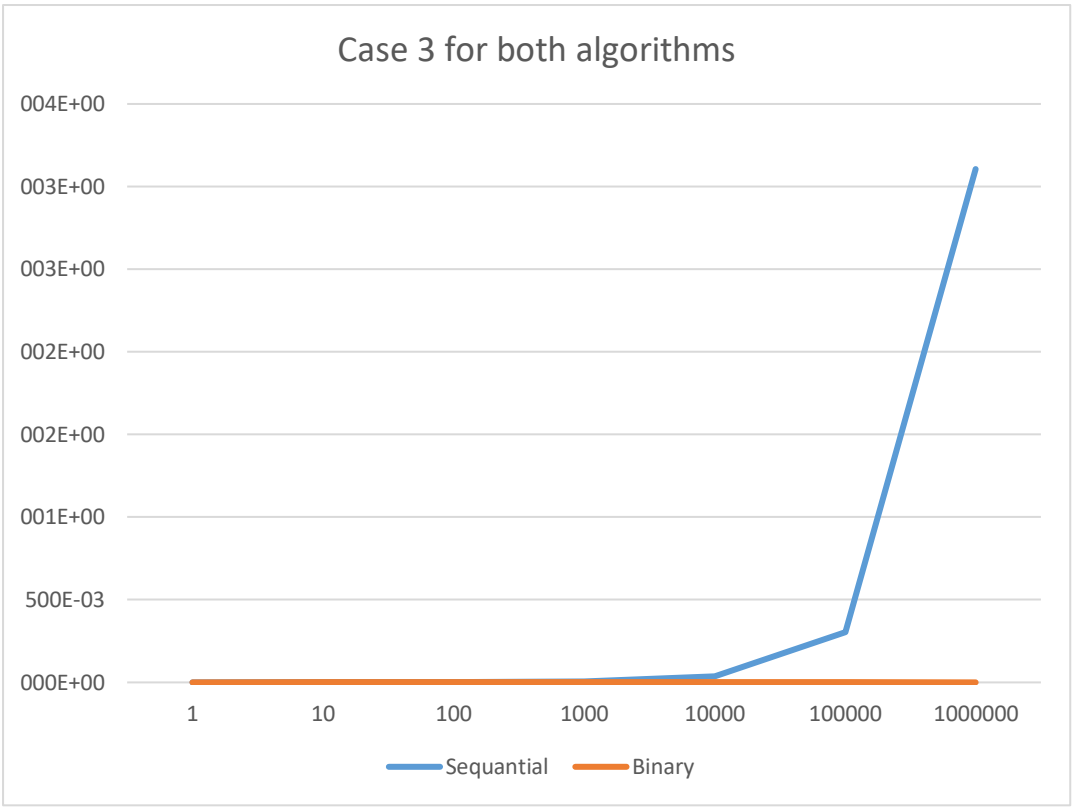
Table for the execution time of sequential search

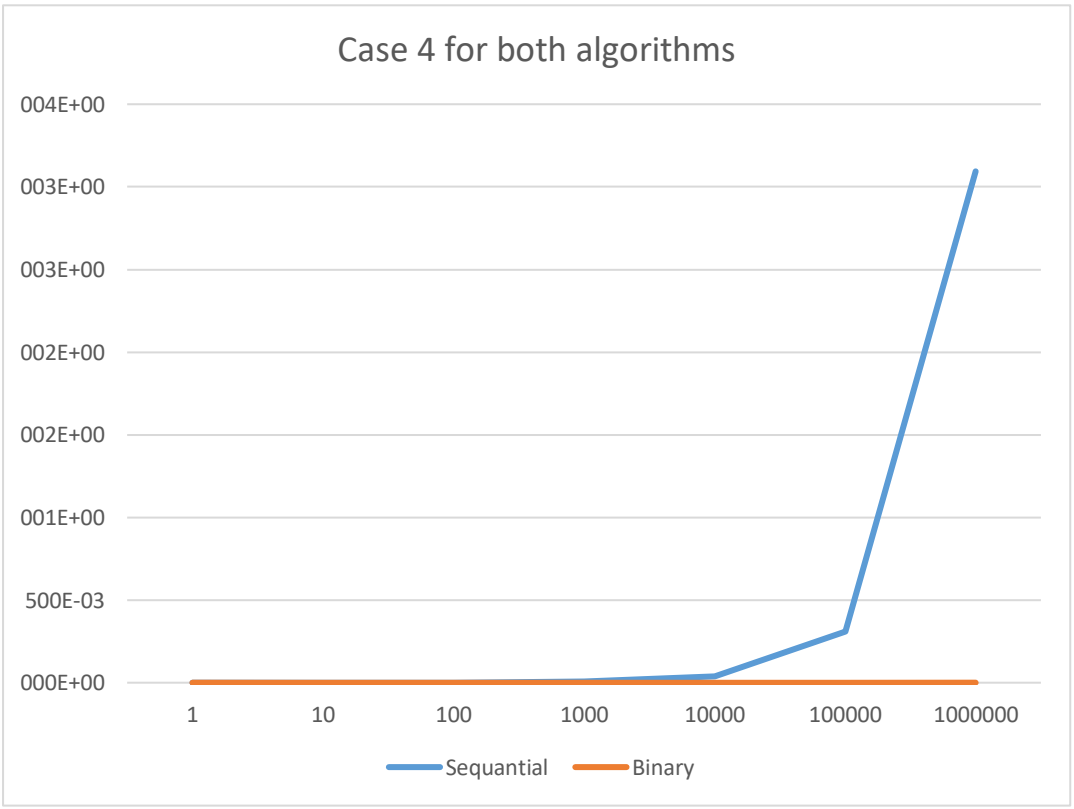
Time (ms)	Case 1	Case 2	Case 3	Case 4
N = 1	6,14e-05	4,16e-05	4e-05	4,6e-05
N = 10	6,07e-05	6,43e-05	6,8e-05	6,45e-05
N = 10^2	8,53e-05	4,59e-05	8,17e-05	8,71e-05
N = 10^3	0,0001023	4,47e-05	0,0001029	0,0001025
N = 10^4	0,0001368	4,37e-05	0,0001276	0,0001296
N = 10^5	0,0001527	4,02e-05	0,0001478	0,0001591
N = 10^6	0,0001925	4,14e-05	0,0001674	0,0001905

Table for the execution time of binary search


Case 1 for both algorithms
0.00025
0.0002
0.00015
0.0001
0.00005
0
1
10
100
1000
10000
100000
1000000
Sequantial
Binary


Case 2 for both algorithms
002E+00
002E+00
001E+00
001E+00
001E+00
800E-03
600E-03
400E-03
200E-03
000E+00
1
10
100
1000
10000
100000
1000000
Sequantial
Binary


Case 3 for both algorithms
004E+00
003E+00
003E+00
002E+00
002E+00
001E+00
500E-03
000E+00
1
10
100
1000
10000
100000
1000000
Sequantial
Binary


Case 4 for both algorithms
004E+00
003E+00
003E+00
002E+00
002E+00
001E+00
500E-03
000E+00
1
10
100
1000
10000
100000
1000000
Sequantial
Binary

In all of the plots, x axis shows the data amount(N) and y axis shows execution time.

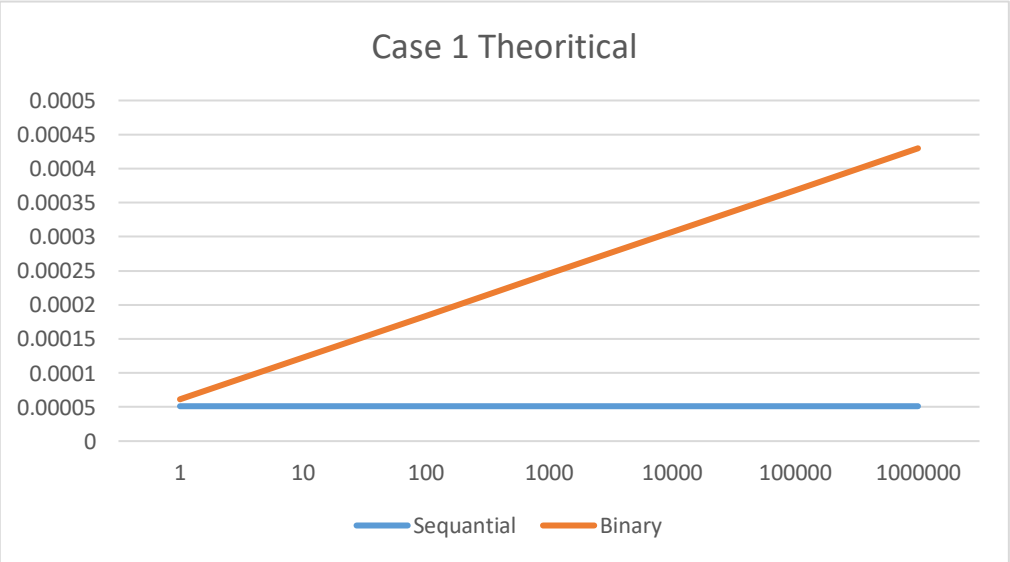
From the plots, we can see that

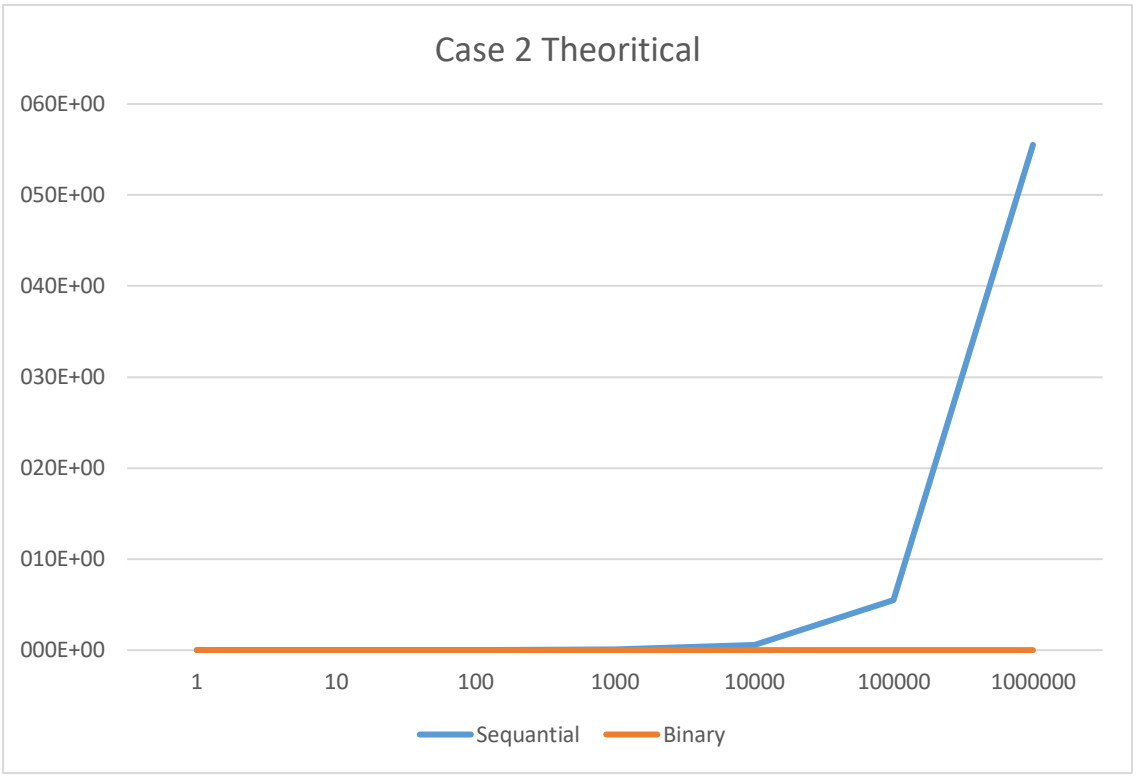
- the best case for sequential search is for key to be the first item of the data, which is in the case 1.
- the best case for binary search is for key to be midpoint of the data, which is in case 2, we can actually physically see this in the table for binary search.
- The worst case for sequential search is either the key doesn't exist or the key is is the last item of the array, while binary search's worst case is the item is in the beginning or at the end of the array or doesn't exist at all.
- The average case for sequential search is the item is in the middle of the array while we can't predict the average case from these plots & tables.

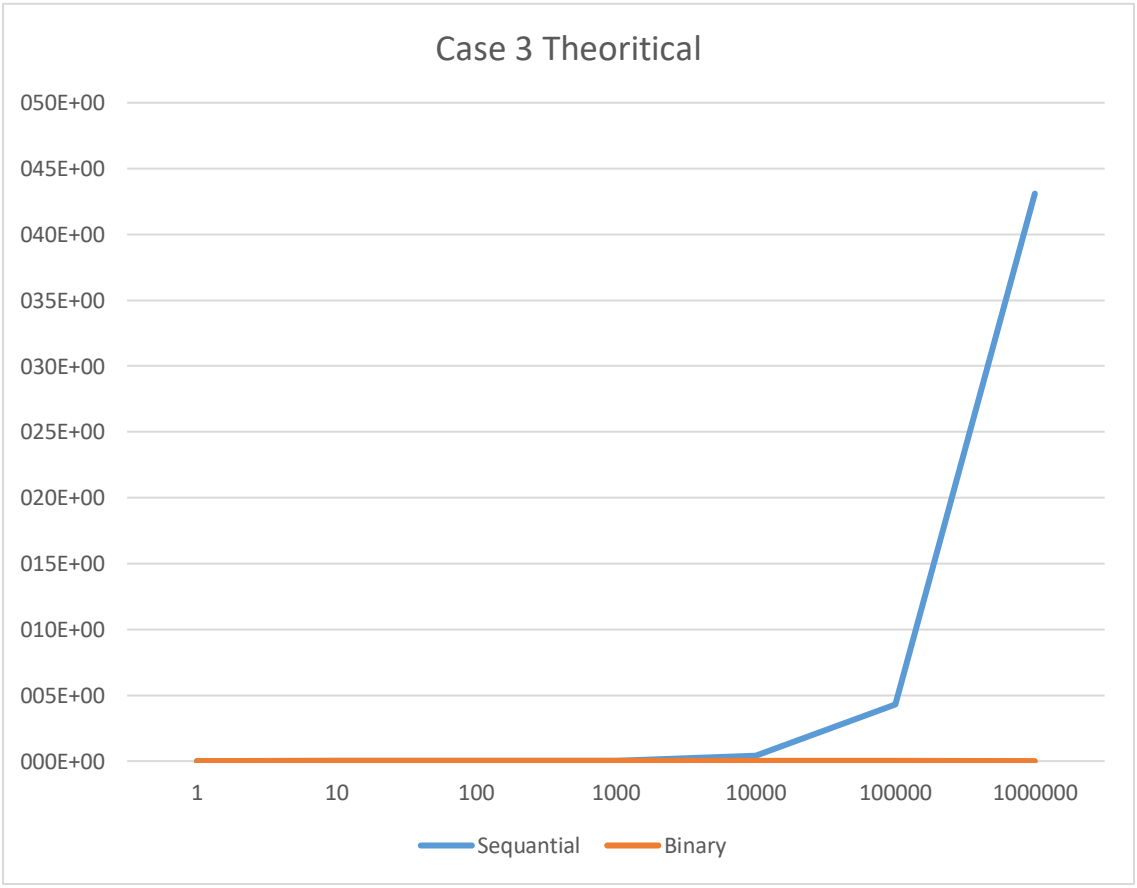
System Specifications

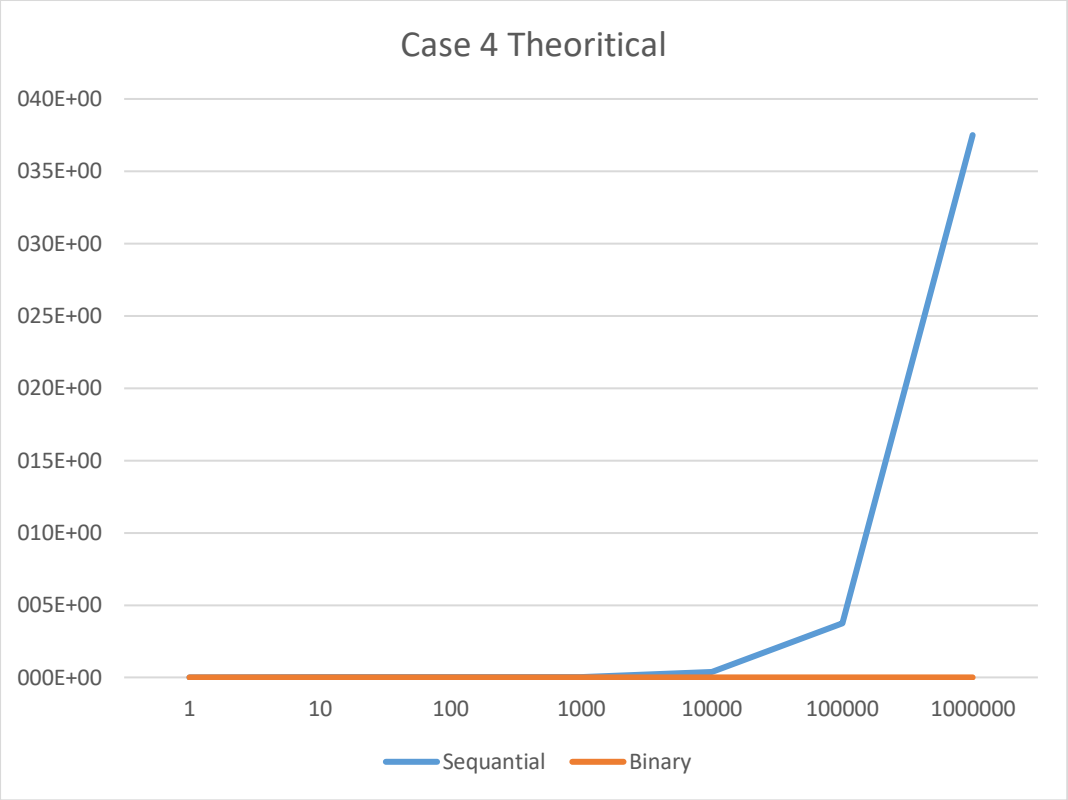
Intel Core i5-5200U @ 2.20 GHz, 8GB RAM, Windows 10, 64-bit operation system.

Following plots are expected results, if we assume that sequential search is O(N) and binary search is O(logN). Same as before, x-axis shows data amount(N) and y-axis shows expected execution time according to data we get when N = 1.




Case 2 Theoritical
060E+00
050E+00
040E+00
030E+00
020E+00
010E+00
000E+00
1
10
100
1000
10000
100000
1000000
Sequantial
Binary


Case 3 Theoritical
050E+00
045E+00
040E+00
035E+00
030E+00
025E+00
020E+00
015E+00
010E+00
005E+00
000E+00
1
10
100
1000
10000
100000
1000000
Sequantial
Binary



Conclusion

As our data grows larger, it is very efficient to use binary search over sequential search if the data is sorted. In small amounts of data, however, one may not need to use binary search since they might find it faster using sequential search, and easier to implement, of course.

In the plotting, we can encounter with some errors if we compare them to the theoretical ones, especially in the case 1. These errors arise due to unstable speed of the computer (e.g. if the computer is doing some other task at the moment), the time computer takes for memory allocation and function calls etc. That's why we don't get straight lines in the plot for case 1, whose data is the smallest.